NASA Technical Memorandum 105869

Inverse Kinematics Problem in Robotics Using Neural Networks

Benjamin B. Choi and Charles Lawrence
Lewis Research Center
Cleveland, Ohio

October 1992

NASA

INVERSE KINEMATICS PROBLEM IN ROBOTICS USING NEURAL NETWORKS

Benjamin B. Choi and Charles Lawrence
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

In this paper, multilayer feedforward networks are applied to the robot inverse kinematic problem. The networks are trained with endeffector position and joint angles. After training, performance is measured by having the network generate joint angles for arbitrary endeffector trajectories. A 3-degrees-of-freedom (DOF) spatial manipulator is used for the study. It is found that neural networks provide a simple and effective way to both model the manipulator inverse kinematics and circumvent the problems associated with algorithmic solution methods.

E-7333

INTRODUCTION

Historically, digital computers have utilized programmed instructions and data patterns to process information and solve problems. Solving a problem by using this programmed approach requires the development of software to implement the algorithms or sets of rules. Frequently, there are situations, such as for nonlinear or complex multivariable systems, where the sets of rules or required algorithms are unknown or too complex to be accurately modeled. Even if characterizing algorithms are obtained, they often are too computationally intensive for practical real-time applications. To circumvent these problems, a relatively new approach to information processing, known as neurocomputing, has been developed. This approach, which does not require algorithm or rule development, is advantageous because it may eliminate software development, decrease computational requirements, and allow for information processing capabilities where algorithms or rules are not known or cannot be derived (refs. 1 to 4).

A neural network is the primary information processing structure of interest in neurocomputing. In reference 1, it is defined as a parallel, distributed information processing structure composed of a number of simple, highly interconnected processing elements similar to neurons in the human nervous system (see fig. 1). The processing elements interact locally through a set of unidirectional weighted connections. The neural network learns (or trains itself) to generalize a mapping or functional relationship from example sets of input vectors and corresponding output vectors. The neural network then stores the connection strengths (weights) between processing units. The weights represent the strength of interconnections between neurons and are adjusted during the learning process. The knowledge of the network is internally represented by the values of the weights and the topology of the connections. From the network's knowledge, the network is able to solve for unknown output when new input is presented. The neural network contrasts with conventional methods where the specific relationships between input and output must be supplied by user defined algorithms or data patterns. The characteristics of neural networks permit self-organization, fault tolerance, generalization, optimization, association, etc. (refs. 1 to 5).

In robotics, the computational requirements for task and path planning, and path control, may be very demanding (refs. 6 and 7). However, most robotic processes may be formulated in terms of optimization or pattern recognition problems so that neural networks can be adapted (refs. 8 to 11). This paper explores the application of a neural network for approximating the nonlinear transformation relating the manipulator's endeffector position to its joint coordinates. From a variety of neural networks, two networks were chosen for the present study: a single-hidden-layer feedforward network, and a multiple-hidden-layer feedforward network. The multilayer feedforward network was chosen because of its

enhanced capability to model nonlinear system characteristics (ref. 2). The networks were trained by using the endeffector position of a 3-degrees-of-freedom (DOF) manipulator as input and its corresponding joint angles as output. After the network was trained, endeffector positions which were not part of the original training data set were input to the network to assess the network's capabilities for generating correct joint angles for arbitrary endeffector positions.

APPLICATION TO MOTION CONTROL OF MICROGRAVITY MANIPULATOR

The relationship between the desired endeffector path in Cartesian space (or work space) and the joint motions (joint space) is defined by the manipulator's kinematics. The determination of the endeffector position from the joint variables is known as forward kinematics. Conversely, inverse kinematics is used to find a set of joint displacements for a given endeffector position. The kinematics and their derivatives are used in path planning and control to provide real-time computing of inverse dynamics and kinematics.

Kinematics of Microgravity Manipulator

Figure 2(a) shows a 4-DOF microgravity manipulator (ref. 12) composed of two intersecting points, each having 2 DOF's. The manipulator has one kinematically redundant DOF if orientation is neglected and only the x, y, z position of the endeffector is specified. Usually, manipulators with redundancy are designed to help overcome kinematic and other limitations such as singularity and obstacle avoidance, joint limits, servomotor torque minimization, and so forth. For the microgravity manipulator, the redundancy is used to reduce the base reactions transmitted into the supporting structure. A cost function, designed to minimize the base forces and moments, is used to resolve the motion of the redundant DOF (ref. 13). In this paper, however, it is assumed that the third joint angle, θ_3, is fixed so that the manipulator is reduced to a 3-DOF, nonredundant configuration. Henceforth, the fourth joint θ_4 is referred to the third joint angle, θ_3, due to sequential order. Since the manipulator is composed of 3 DOF's, there are four possible solution sets (see eqs. (5), (6), and (7)) to the inverse kinematics for a given endeffector position. These correspond to left/right shoulder and up/down elbow configurations. To simplify the problem, the network is trained with only one (right shoulder and up elbow) of the multiple solutions.

For the forward kinematics of the microgravity manipulator, the joint parameters (θ_1, θ_2, θ_3) are mapped to endeffector position (x, y, z). On the basis of figure 2, the Denavit-Hartenberg (D-H) link parameters of the manipulator are obtained in terms of the link frames and are listed in table I. From these values (the values of joint limit ranges differ from those in ref. 12), the D-H homogeneous transformation matrices (refs. 7 and 14) that are the 4 x 4 transformation matrices relating rotation and displacement between the two coordinate systems are defined as:

$$T_1^0 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$T_2^1 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$T_3^2 = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & L_1 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where T_i^{i-1} is a transformation matrix that relates coordinate system i to i-1. The Cartesian coordinates of the endeffector in the base system can be expressed as:

$$\begin{bmatrix} p^0 \\ 1 \end{bmatrix} = T_1^0 T_2^3 T_3^2 \begin{bmatrix} p^3 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} L_1\cos\theta_1\cos\theta_2 + L_2\cos\theta_1\cos(\theta_2+\theta_3) \\ L_1\sin\theta_1\cos\theta_2 + L_2\sin\theta_1\cos(\theta_2+\theta_3) \\ L_1\sin\theta_2 + L_2\sin(\theta_2+\theta_3) \\ 1 \end{bmatrix} \tag{4}$$

The inverse kinematic problem involves mapping the endeffector position (x, y, z) to the joint parameters (θ_1, θ_2, θ_3). Unlike the forward kinematic problem, the inverse kinematic problem usually does not have a unique solution. Several joint positions may provide the identical endeffector position. Moreover, if the system has a greater number of joints (DOF's) than the minimum number necessary for the task, the system will be redundant (refs. 15 and 16), and will have an infinite number of solutions. A variety of procedures, such as the matrix algebraic method, zero position method, geometric inspection method, and so on (refs. 6, 14, 17, and 18), have been developed for solving the kinematic problem. For complex systems, however, it is sometimes very difficult, or even impossible, to obtain a set of closed-form solutions. Moreover, many kinematic solutions which are not closed-form require iterative procedures such as the Newton Raphson or the gradient descent methods (ref. 16). These numerical methods are very time consuming and have severe limitations for real-time operations.

The angle θ_1, shown in figure 2(b), can be obtained as

$$\theta_1 = ATAN2(p_y, p_x) \tag{5}$$

Applying the law of cosine to the geometry gives

$$\theta_3 = \pm \cos^{-1}\left(\frac{p_x^2 + p_y^2 + p_z^2 - L_1^2 - L_2^2}{2 L_1 L_2}\right) \tag{6}$$

and

$$\theta_2 = \gamma - \psi \tag{7}$$

where

$$\gamma = ATAN2\left(p_z, \sqrt{p_x^2 + p_y^2}\right)$$

and

$$\psi = \pm \cos^{-1}\left(\frac{p_x^2 + p_y^2 + p_z^2 + L_1^2 - L_2^2}{2 L_1 \sqrt{p_x^2 + p_y^2 + p_z^2}}\right)$$

Equations (5), (6), and (7) represent the inverse kinematics problem of the 3-DOF microgravity manipulator. A solution set of right shoulder and up position provides the joint angles used for training the network. Once the network is trained, its capability for predicting subsequent joint angles is measured by comparing the network's solutions to those provided by the closed-form solutions of equation (4).

Network Architecture

A unique characteristic of the neural network architecture is that it is configured by the user and can be easily tailored to fit the application. A question arises concerning the best network architecture for a particular application. Factors to consider are the number of hidden layers, the number of neurons in each hidden layer, how those layers are connected to each other, the number of training data points, how those points are selected, and so on. Typically, these decisions are based on the designer's experience. It is important to note that the computational requirements are independent of the number of DOF's in the manipulator; instead, they are based on the network architecture. In this study, the PC based ANZA processing board (ref. 19) and the CRAY based NETS 2.01 software (ref. 20) were used for the single-hidden-layer and the multi-hidden-layer networks, respectively.

Let us consider first a single-hidden-layer network for a two-dimensional case. Figure 3 shows the simplest network architecture used for the present study. The input layer distributes the endeffector y and z coordinates to the connections which then weigh the coordinate values and distribute them to the hidden layer. In the hidden layer, each neuron transforms the accumulated incoming data and passes the result through a sigmoid transformation function. The transformed data then is transported to the output layer through another arrangement of weighted connections. The neurons in the output layer correspond to the θ_2 and θ_3 joint angles. The final result of weighting, transformation, and distribution of data is that the input data is transformed (or mapped) into corresponding output data in the output layer. For the three-dimensional manipulator, the 2-3-2 network of figure 3 is expanded to the 3-3-3 network that includes three input neurons x, y, and z for the endeffector position; three neurons in the hidden layer; and three output neurons, θ_1, θ_2, and θ_3 for the joint angles.

During the learning procedure, the errors between the actual network output and the training output are minimized by updating the weights. The updating scheme is implemented so that changes recursively propagate back through the network, changing weights that had a large effect on the output more than those which did not. This process is repeated until the training and computed output are within a predefined training error (see ref. 2 for details).

In this study, two measures are used to evaluate prospective networks. The first measure is termed *training error*. This measure is defined as the root-mean-squared (RMS) difference between the network output and the training point data. Normally, the training error is predefined by the user, and an iterative process which updates the weights is repeated until the error requirements are met.

The second measure is termed *performance error* and is used to evaluate the network's ability to generate accurate output after the training is completed. This measure is defined as the RMS difference between the actual network output and the correct joint angles for data points which are not a part of the original training points.

SIMULATION RESULTS AND DISCUSSION

Two-Dimensional Case

The 2-3-2 network architecture (fig. 3) was used for the first attempt to model the manipulator's kinematics. As previously mentioned, the two neurons in the input layer correspond to the y-z coordinates of the endeffector, there are three neurons in the hidden layer and two neurons in the output layer corresponding to the joint angles θ_2 and θ_3. The input layer was completely connected to the hidden layer, and the hidden layer was also completely connected to the output layer. There were no direct connections from the input to output layers. For the two-dimensional case, θ_1 is fixed, and joint angles θ_2 and θ_3 are free to rotate in the y-z plane.

Figure 4 shows the microgravity manipulator workspace. The y-z coordinates were scaled to between 0.1 and 0.9 (ref. 20) to utilize the effective range of the sigmoid transfer function. Similarly, coordinates θ_2 and θ_3 also were scaled.

As a start, the network was trained with 100 randomly chosen training points comprised of 100 endeffector positions and the corresponding joint angles. For this system, a 15.4-percent training RMS accuracy was obtained after 200 iterations. Additional iterations did not improve the training error because of the limitations of the 2-3-2 network. It was not expected that the network could adequately perform with this limited network architecture.

To improve the network's training error, four additional network architectures having 6, 12, 50, and 100 neurons in the hidden layer were tested to determine the trade-off between the size of the hidden layer and training error (fig. 5). Up to the 2-50-2 network, the networks with more neurons in the hidden layer were able to produce smaller errors with fewer iterations. For a 2-100-2 network, however, the training error could not be improved further, and more iterations were required to obtain the same training error as for the 2-50-2 network. The larger networks, due to their number of neurons and connections, did require more training time for each iteration. To provide a reasonable balance between training error and computation effort, a 2-20-2 network architecture was selected for training.

Now, it is interesting to determine the minimum number of training points required to characterize the inverse kinematics. At least three training points are necessary to define a two-dimensional plane.

Hence, it may be a reasonable start to train a 2-20-2 network with three training points. However, a network was trained with only a single training point to verify this assumption empirically. Fifty-three iterations were required to attain a 1-percent RMS training error. After the training was completed, the network was evaluated by having it attempt to produce the joint angles for a circular endeffector trajectory. The circle was composed of 50 data points. None of the data points were a part of the original training set.

Figure 6(a) compares the correct joint angles with those generated by the network which was trained with only one training point. The joint angles produced by the network were input into the forward kinematics equations to compute the endeffector position. Figure 6(b) shows the correct and actual endeffector positions. As expected, the network was incapable of producing accurate joint angles even though it was able to reproduce the sole training point. Figure 7 shows results from a network trained with two training points. For this network, joint angle θ_3 follows the actual trajectory more closely than angle θ_2. However, this network also is unable to produce accurate output for the circular trajectory. Figure 8 shows results from the network trained with three training points. With three training points, the network is much more able to generate the joint angles corresponding to the desired circular trajectory (7-percent RMS accuracy). Even further improvement (5-percent RMS accuracy) is shown by training the network with four training points (fig. 9).

Figure 10 shows the required number of iterations for a 1-percent RMS training error as a function of the number of training points. While increasing the number of training points improves the network's performance, it also increases the number of iterations required for a specified training error. The number of iterations is increased because the weights must satisfy the specified error for all of the training points simultaneously. The general trend is for the number of iterations to increase exponentially with the number of training points. For five or more training points, the 2-20-2 network could not be trained within the 1-percent RMS accuracy, regardless of the number of iterations.

Figure 11 shows the network's ability to produce five desired circular trajectories. The network was trained with the four points shown in figure 9. The portions of the circular trajectories, which are in the region of the training points, are reproduced very well. The trajectories further from the training points are not reproduced as well.

In order to increase the network performance, a multilayered network with two hidden layers (2-10-10-2) was investigated. For this network, 73 training points were used, so that the entire two-dimensional manipulator workspace was represented (fig. 12). For a 1-percent RMS training accuracy, the network was able to reproduce the training points with 150,000 iterations. Note that the 2-20-2 network could not even reproduce five data points with this same accuracy.

Figure 13 shows the 2-10-10-2 network's capabilities for producing circular trajectories. The trajectories were the same ones used for assessing the 2-20-2 performance (fig. 10). Clearly, the 2-10-10-2 network is able to accurately generate joint angles for endeffector positions anywhere within the workspace. In fact, the joint angle accuracy is within 1 percent RMS for any of the trajectories. It is interesting to note that the 2-20-2 network, which has the same number of neurons as the 2-10-10-2 but has only one hidden layer, does not perform nearly as well.

Three-Dimensional Case

For the three-dimensional case, three neurons are required for both the input and output layers, corresponding to the x, y, and z and θ_1, θ_2, and θ_3 coordinates, respectively. To accommodate these requirements and acquire the advantages of the multilayer network, a 3-10-10-3 system is used.

Figure 14 shows the desired joint angles for a circular trajectory and the angles generated by the network which was trained with four training points. Also shown are the desired and generated endeffector positions. Obviously, the network does not adequately perform after being trained with only four points. Figure 15 illustrates the network performance after being trained with five points. For this particular trajectory, it appears that five training points are sufficient for characterizing the inverse kinematics. The results of figure 16 were created by training the network with 27 training points, then having it generate four arbitrary trajectories. For each of the four trajectories, the network was able to map the trajectory to within 2-percent RMS accuracy.

CONCLUDING REMARKS

In this paper, several neural network architectures were evaluated for modeling the inverse kinematics of a two- and three-dimensional manipulator. The following conclusions are drawn from this study.

1. With adequate training, the neural network may provide a practical and efficient solution to the inverse kinematic problem of robot manipulators.

2. The neural network is able to infer accurate solutions to the inverse kinematics from limited training data.

3. The network with two hidden layers out-performed the single-layer network even when both had the same total number of neurons.

4. The location and quantity of training points is dependent on the manipulator workspace and the desired trajectories.

5. Considering the success of the neural network for resolving the inverse kinematic problem, future studies may address the application of neural networks for solving the redundant manipulator kinematic problem where minimal base reactions are desired.

ACKNOWLEDGMENTS

The authors would like to thank P. Baffes of the NASA Lyndon B. Johnson Space Center and L. Berke of the NASA Lewis Research Center for the development of the publicly available NETS 2.01 software.

REFERENCES

1. Hecht-Nielsen, R.: Neurocomputing. Addison-Wesley Publishing Co., Reading MA, 1991.

2. Rumerlhart, D.E.; and McClelland, J.L.: Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. 1. MIT Press, Cambridge, MA, 1986.

3. Soucek, B.; and Soucek, M.: Neural and Massively Parallel Computers: The Sixth Generation. John Wiley & Sons, Inc., New York, 1988.

4. Anderson, J.A.; and Rosenfeld, E.: Neurocomputing: Foundations of Research, MIT Press, Cambridge, MA, 1989.

5. Berke, L.; and Hajela, P.: Applications of Artificial Neural Nets in Structural Mechanics, NASA TM–102420, 1990.

6. Shin, K.G.; and Malin, S.B.: A Structural Framework for the Control of Industrial Manipulator. IEEE Trans. Syst., Man, Cybernetics, vol. SMC–15, Jan./Feb. 1985, pp. 78–94.

7. Craig, J.J.: Introduction To Robotics: Mechanics and Control. Addison-Wesley Publishing Co., Reading, MA, 1989.

8. Kung, S.Y.; and Hwang, J.N.: Neural Network Architectures for Robotics Applications. IEEE Trans. Rob. Autom., vol. 5, no. 5, Oct. 1989, pp. 641–657.

9. Thorp, C.E.: Path Relaxation: Path Planning for a Mobile Robot. Proceedings of the National Conference on Artificial Intelligence, AAAI, Menlo Park, CA, 1984, pp. 318–321.

10. Sun, G.Z.; Chen, H.H.; and Lee, Y.C.: Neural Network Representation of Sensor Graphics for Autonomous Robot Navigation. Proceedings of First IEEE International Conference on Neural Networks, M. Caadill and C. Butler, eds., San Diego, CA, Vol. 4, IEEE, New York, 1987, pp. 345–356.

11. Ellison, D.: On the Convergence of the Multidimensional Albus Perceptron, Int. J. Rob. Res., vol. 10, no. 4, Aug. 1991, pp. 338-357.

12. Herndon, J.N., et al.: Manipulator Hardware for Microgravity Research. Proceedings of 38th Conference on Remote Systems Technology, G. Fradsen, ed., Vol. 2, American Nuclear Society, La Grange Park, IL, 1990, pp. 161–168.

13. Chung, C.L.; and Desa, S.: A Global Approach for Using Kinematic Redundancy to Minimize Base Reactions of Manipulators. Report CMU–RI–TR–89–9. Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, 1989.

14. Denavit, J.; and Hartenberg, R.S.: A Kinematics Notation for Lower-Pair Mechanisms Based On Matrices. J. Appl. Mech., Vol. 22, June 1955, pp. 215–221.

15. Benhabib, B.; Goldenberg, A.A.; and Fenton, R.G.: A Solution to the Inverse Kinematics of Redundant Manipulators. J. Rob. Syst., vol. 2, no. 4, 1985, pp. 373–385.

16. Chang, P.H.: A Closed-Form Solution for Inverse Kinematics of Robot Manipulators with Redundancy. IEEE J. Rob. Autom., vol. RA–3, no. 5, Oct. 1987, pp. 393–403.

17. Gupta, K.C; and Roth, B.: Design Considerations for Manipulator Workspace. J. Mech. Design, vol. 104, 1982, pp. 704–711.

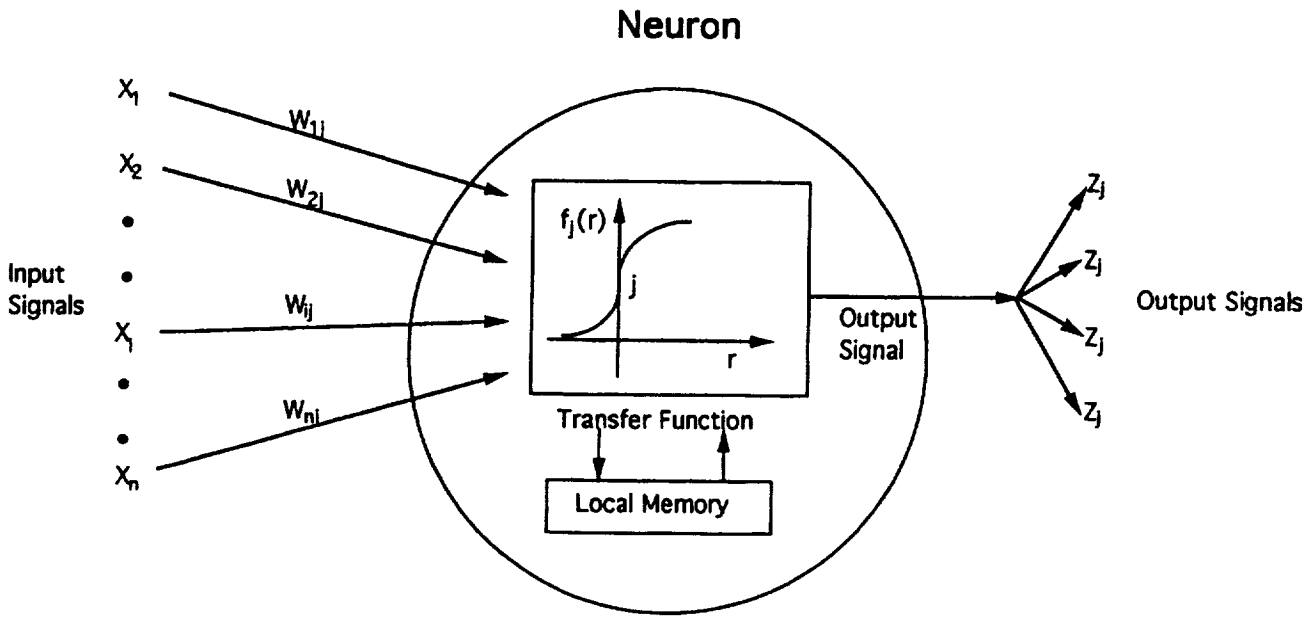
18. Litvin, F.L.: Simplification of the Matrix Method of Linkage Analysis by Division of a Mechanism into Unclosed Kinematic Chains. Mech. Machine Theory, vol. 10, 1975, pp. 315–326.

19. ANZA User's Guide. Hecht-Nielsen Neurocomputer Corp., San Diego, CA, 1988.

20. Baffes, P.T.: Nets User's Guide, Software Technology Branch, NASA/Lyndon B. Johnson Space Center, Houston, TX, Sept. 1989.

TABLE I.—DENAVIT-HARTENBERG LINK PARAMETERS OF THE MICROGRAVITY MANIPULATOR

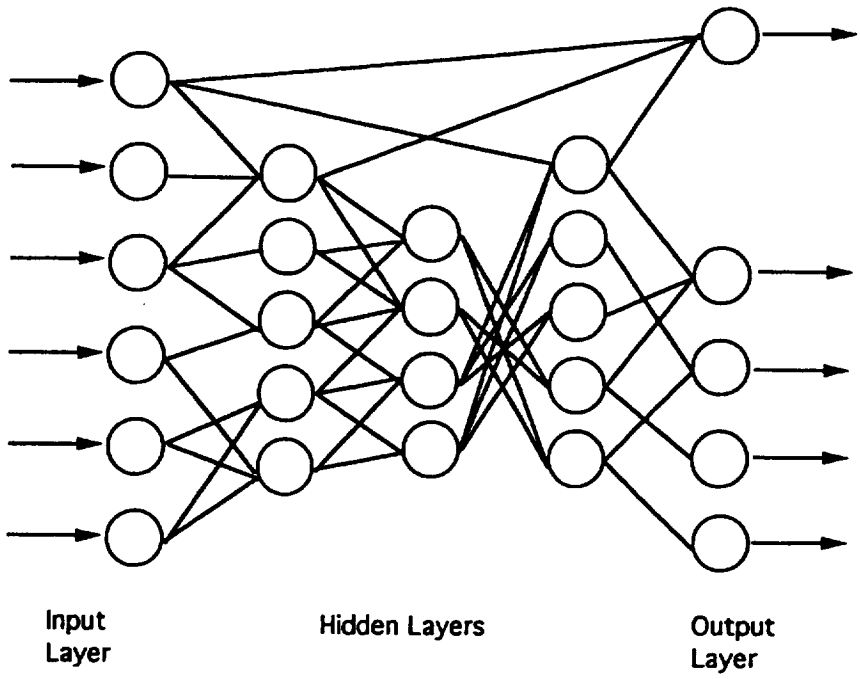
i	α_{i-1}, deg	a_{i-1}	d_i	θ_i	Range, deg
1	0	0	0	θ_1	$-180 < \theta_1 < 180$
2	90	0	0	θ_2	$-45 < \theta_2 < 120$
3	0	20	0	θ_3	$-30 < \theta_3 < 135$



$$r_j = \sum_i X_i W_{ij}$$

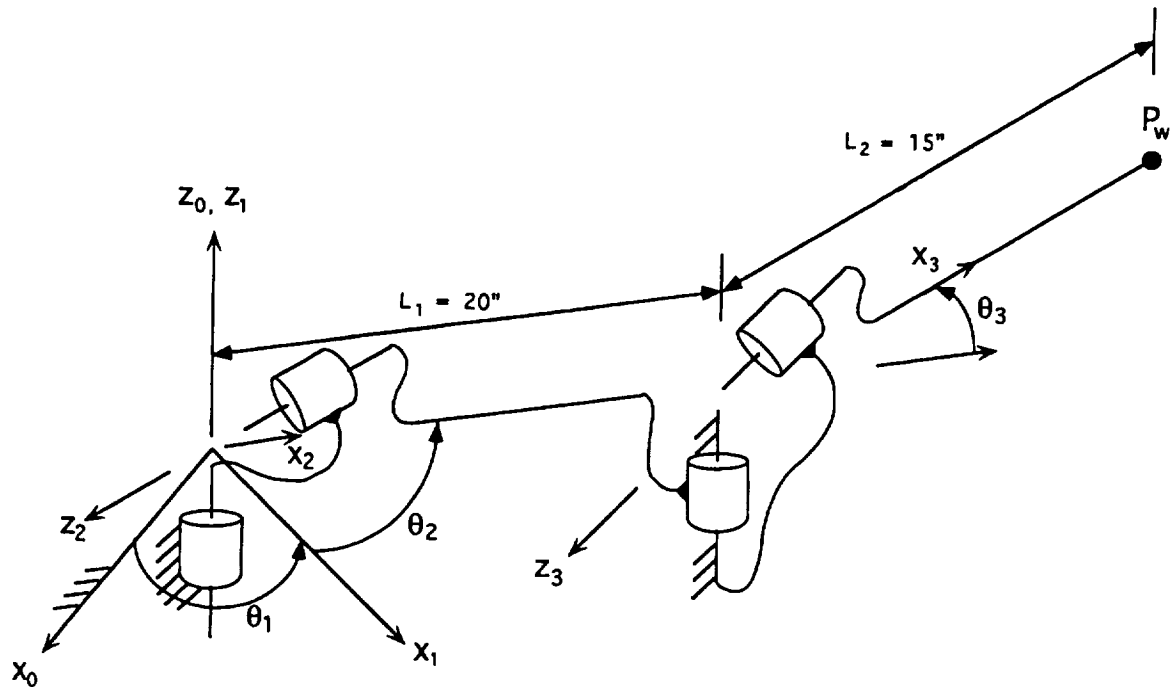
$$Z_j = f_j(r)$$

(a) Generic artificial neuron.

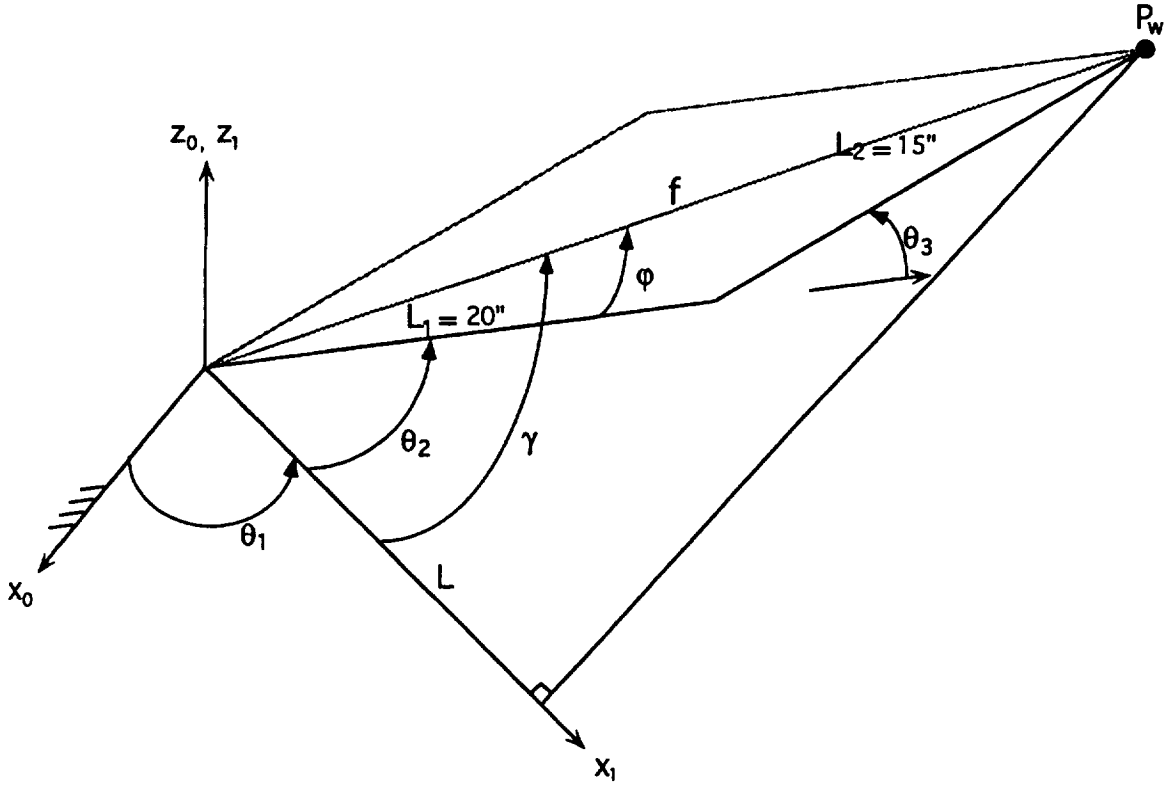


(b) Generic neural net.

Figure 1.—Neural network architecture.



(a) Kinematic parameters and coordinate systems for the microgravity manipulator.



(b) Geometric description of (a).

Figure 2.—Schematic drawing of microgravity manipulator.

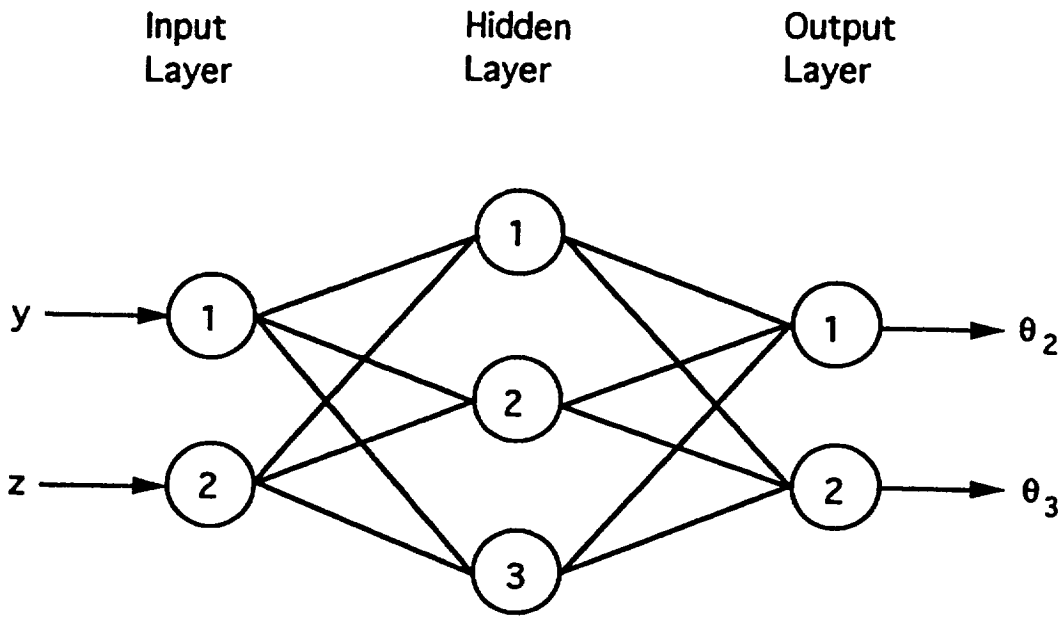


Figure 3.—Model of 2 - 3 - 2 network (2 neurons in input layer, 3 neurons in hidden layer, and 2 neurons in output layer).

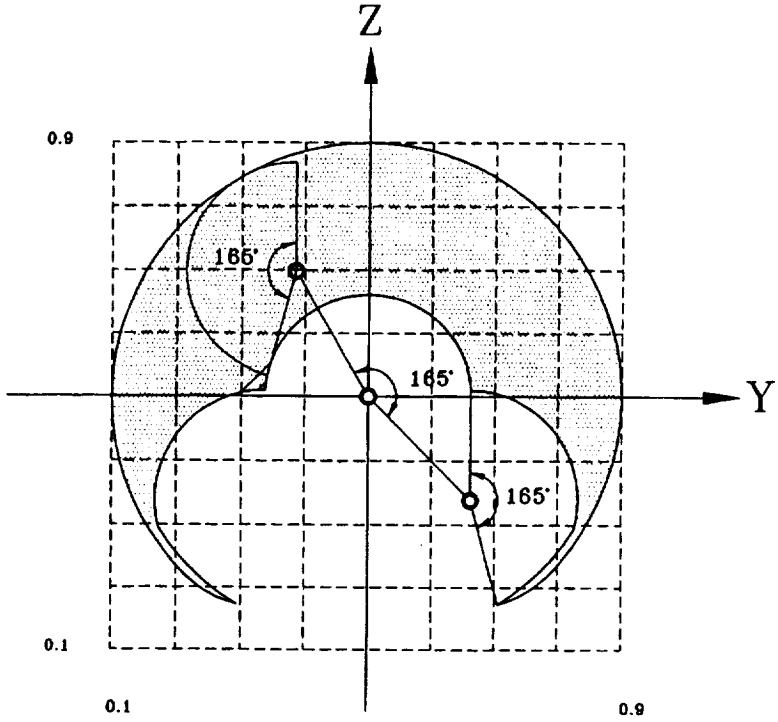


Figure 4.—Cross section of the microgravity manipulator workspace in y-z plane.

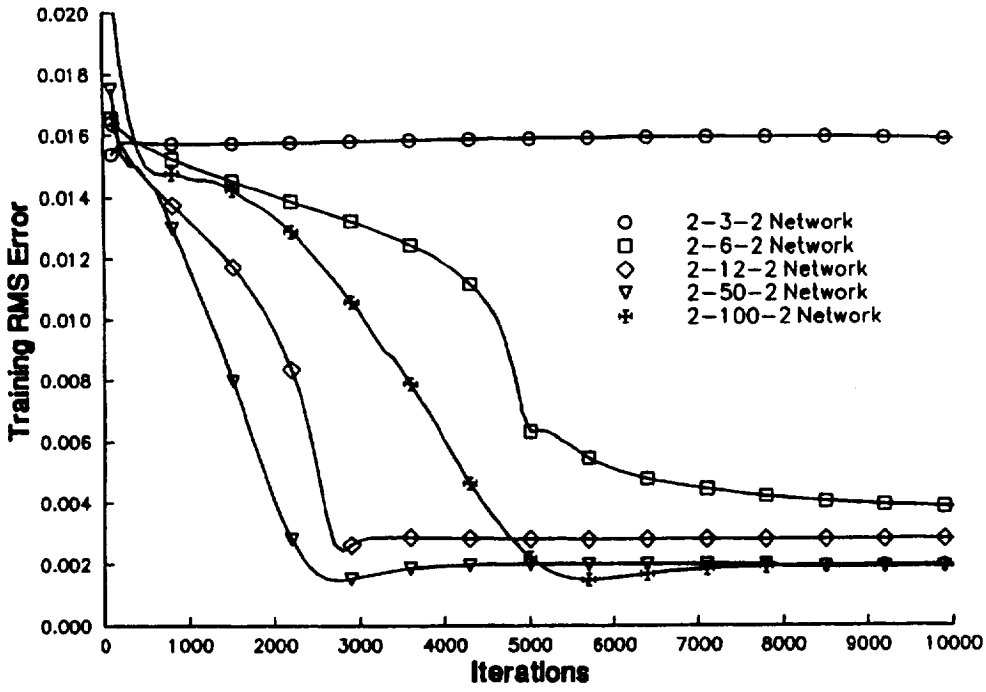
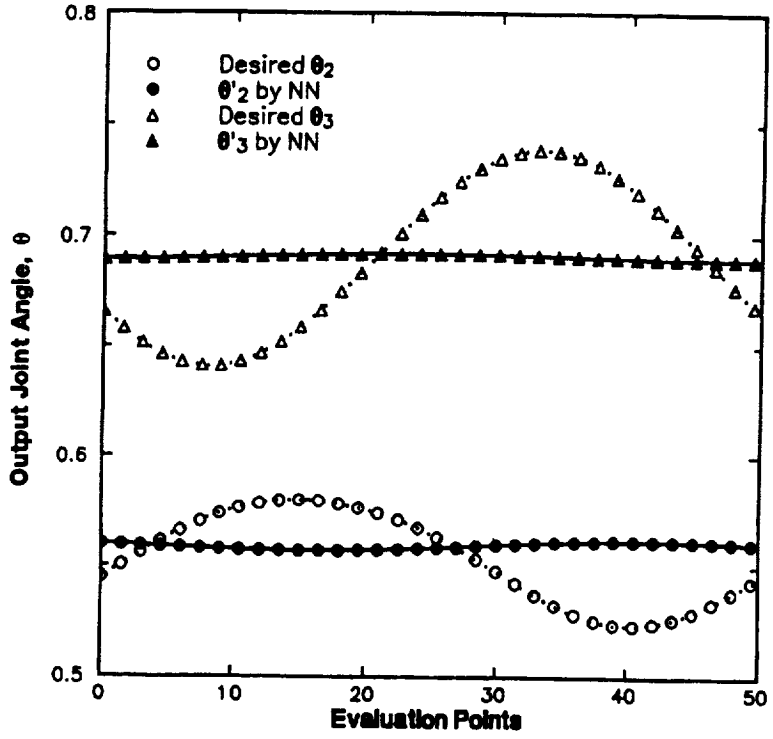
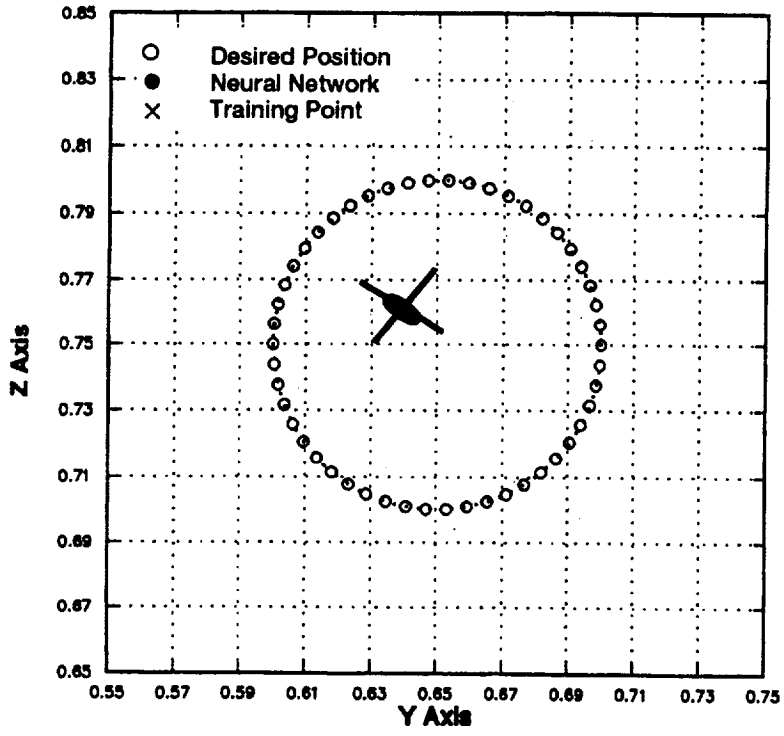


Figure 5.—Error convergence of neural network architectures using 100 randomly chosen training points.

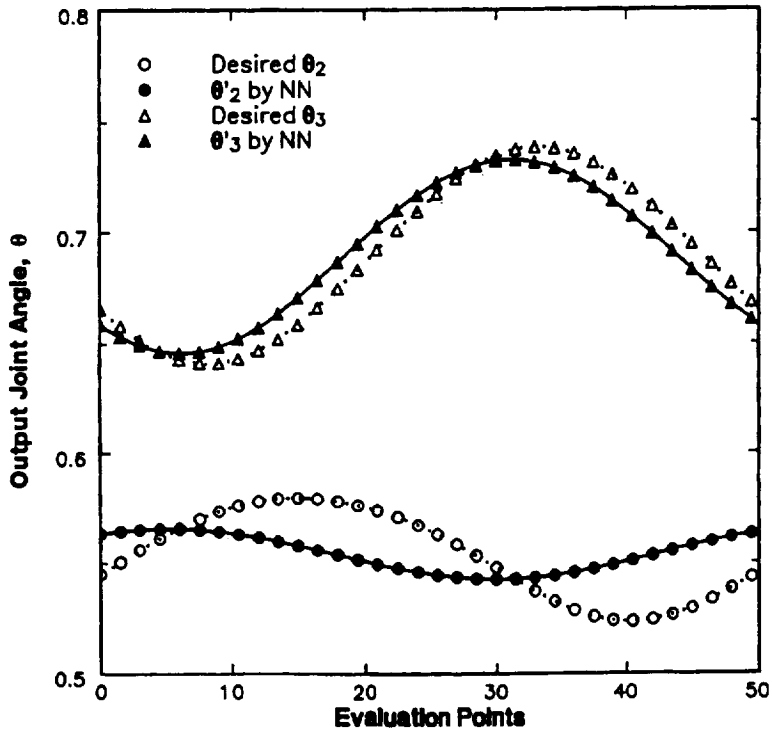


(a) Joint angles.

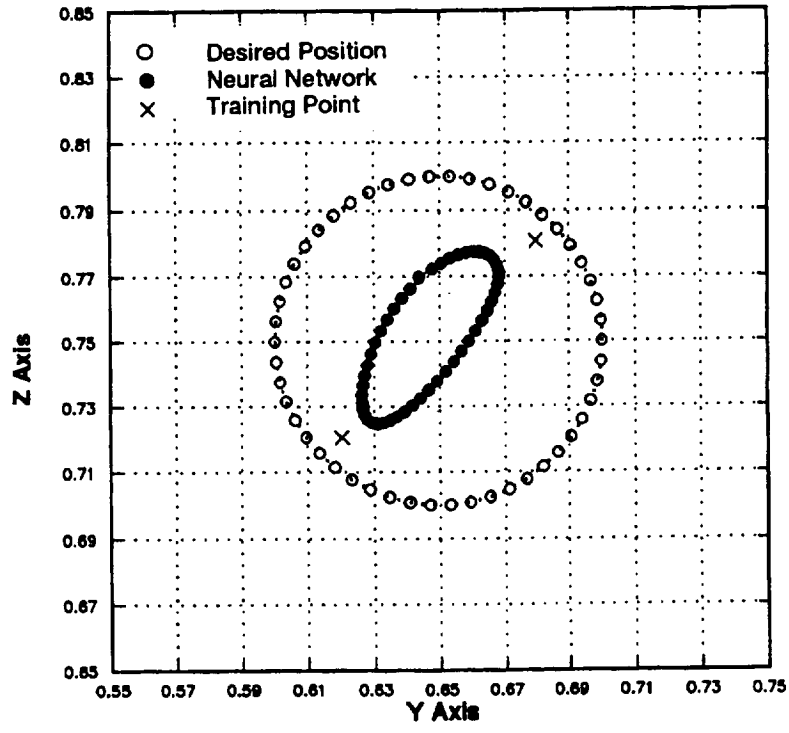


(b) Endeffector positions.

Figure 6.—Results from a neural network using one training point.

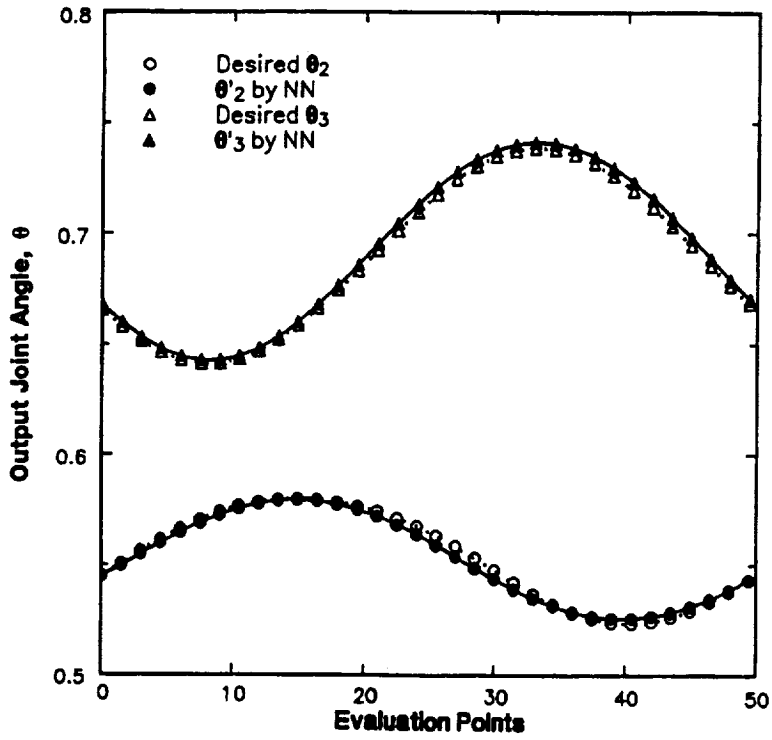


(a) Joint angles.

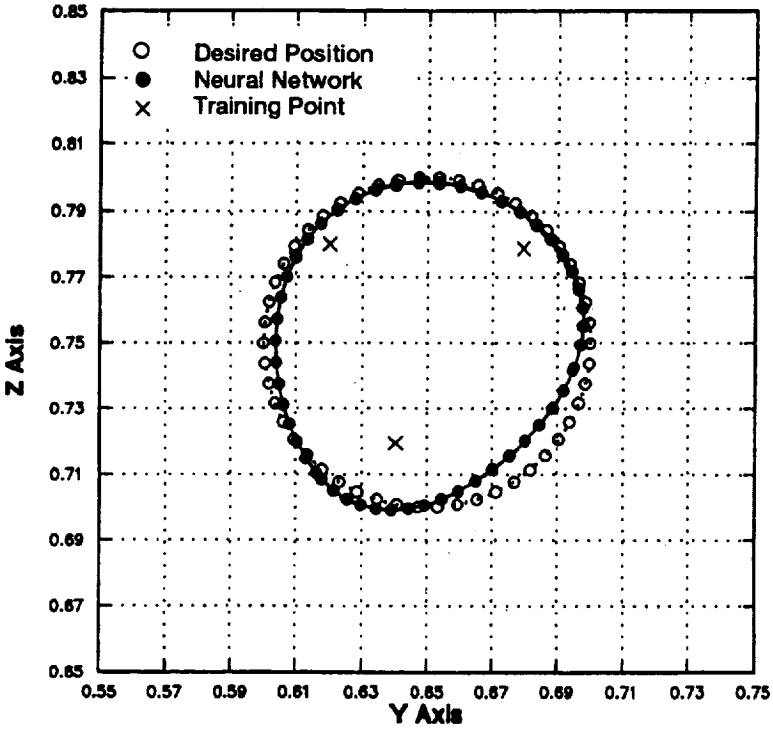


(b) Endeffector positions.

Figure 7.—Results from a neural network using two training points.

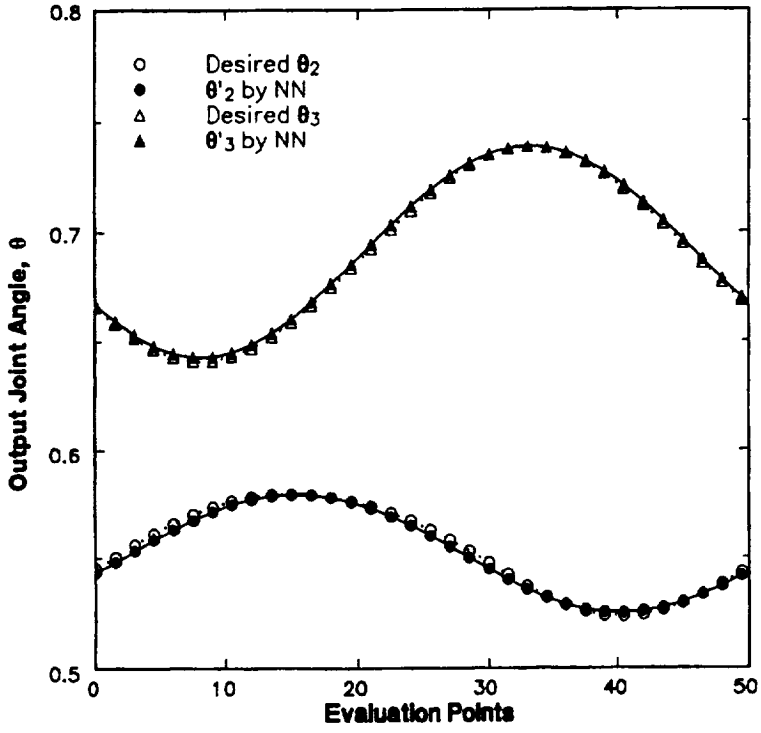


(a) Joint angles.

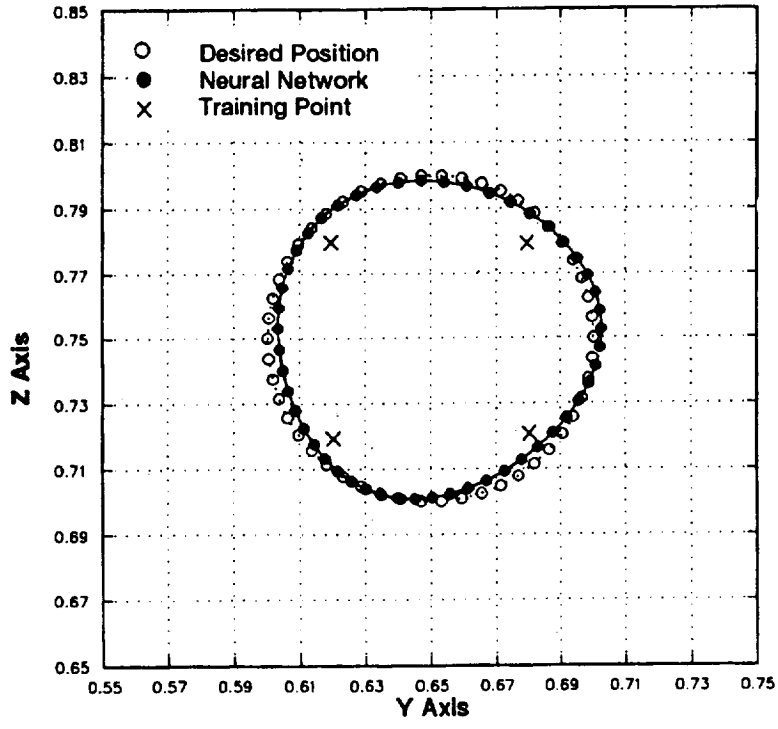


(b) Endeffector positions.

Figure 8.—Results from a neural network using three training points.



(a) Joint angles.



(b) Endeffector positions.

Figure 9.—Results from a neural network using four training points.

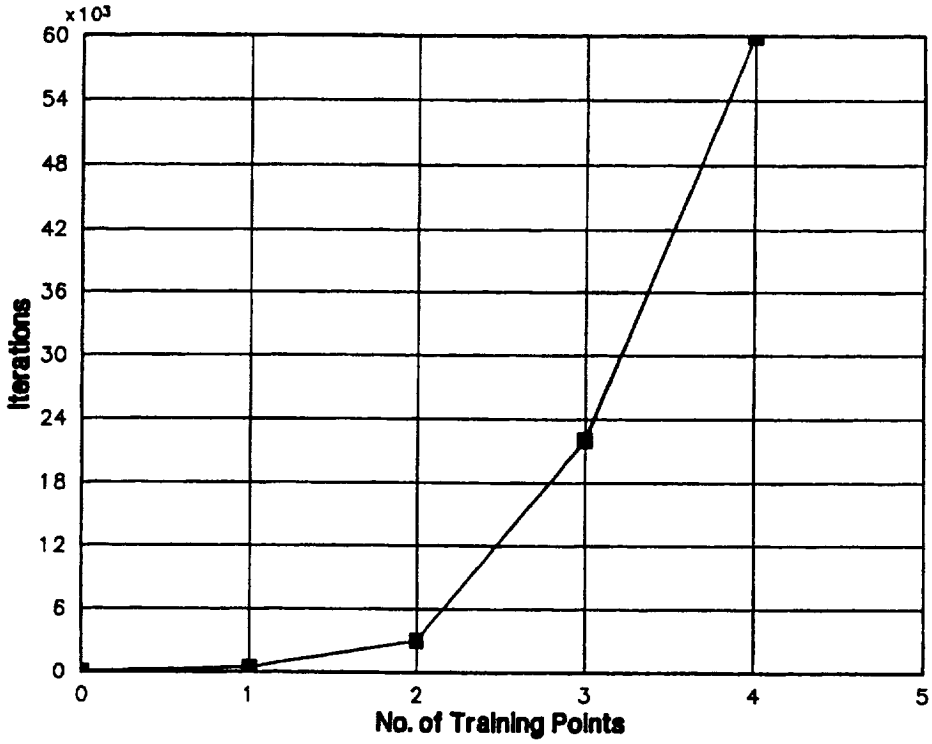


Figure 10.—Training iterations for 1-percent RMS accuracy.

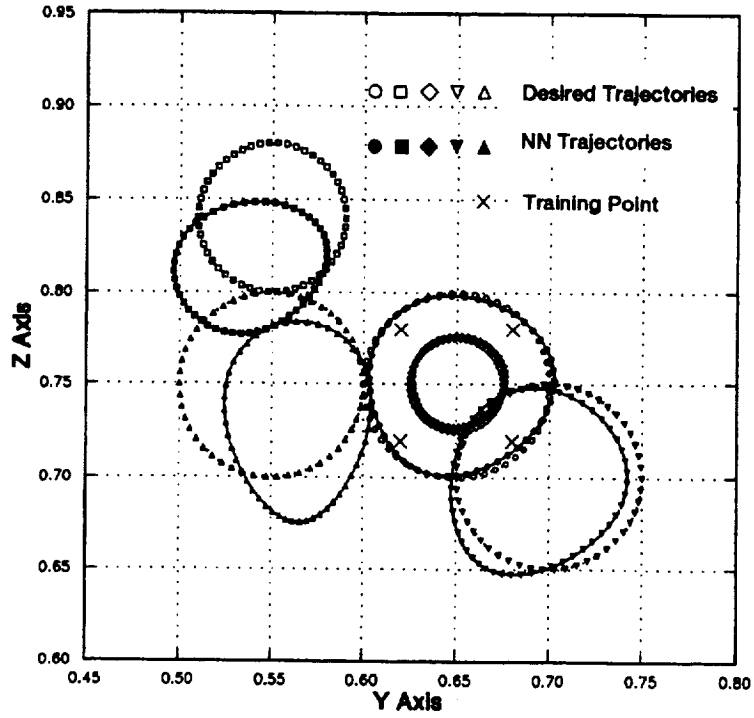


Figure 11.—A 2-20-2 network using four training points.

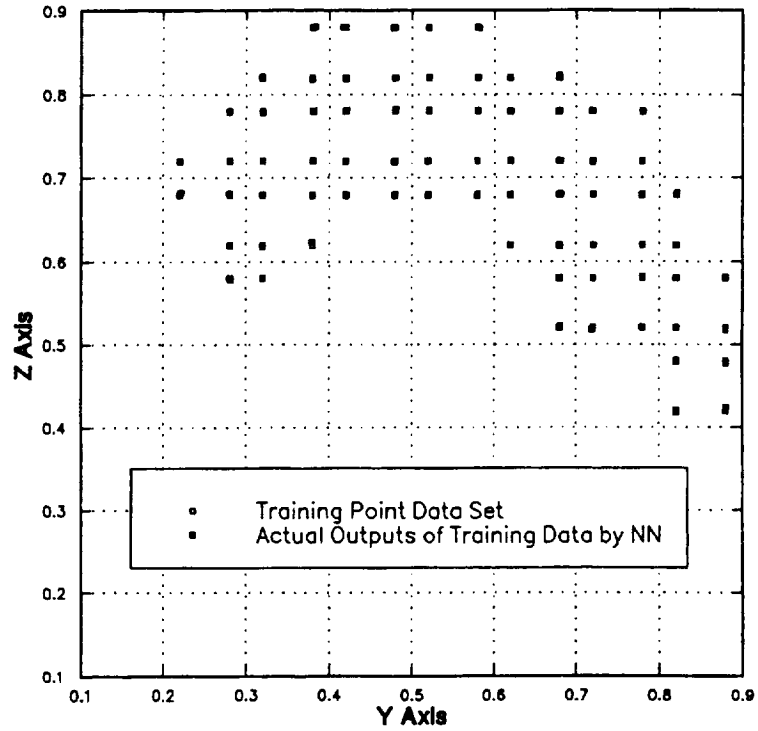


Figure 12.—Seventy-three network training points.

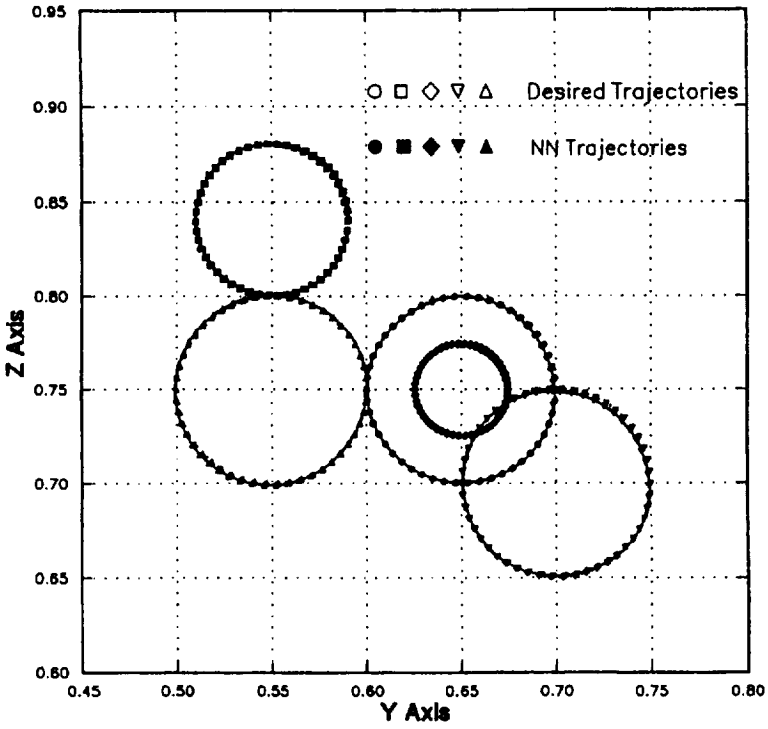
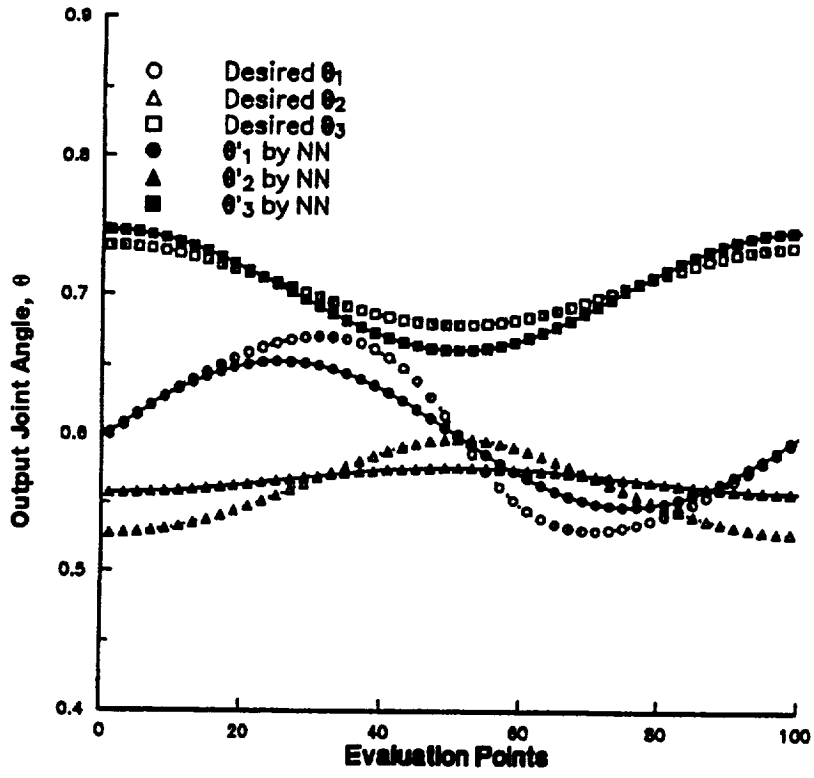
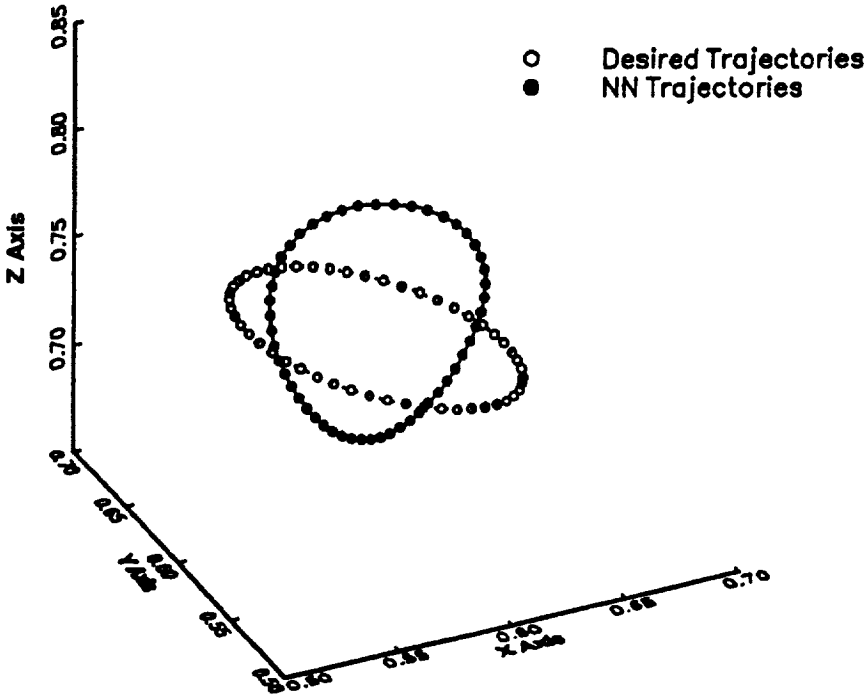


Figure 13.—A 2-10-10-2 network using 73 training points.

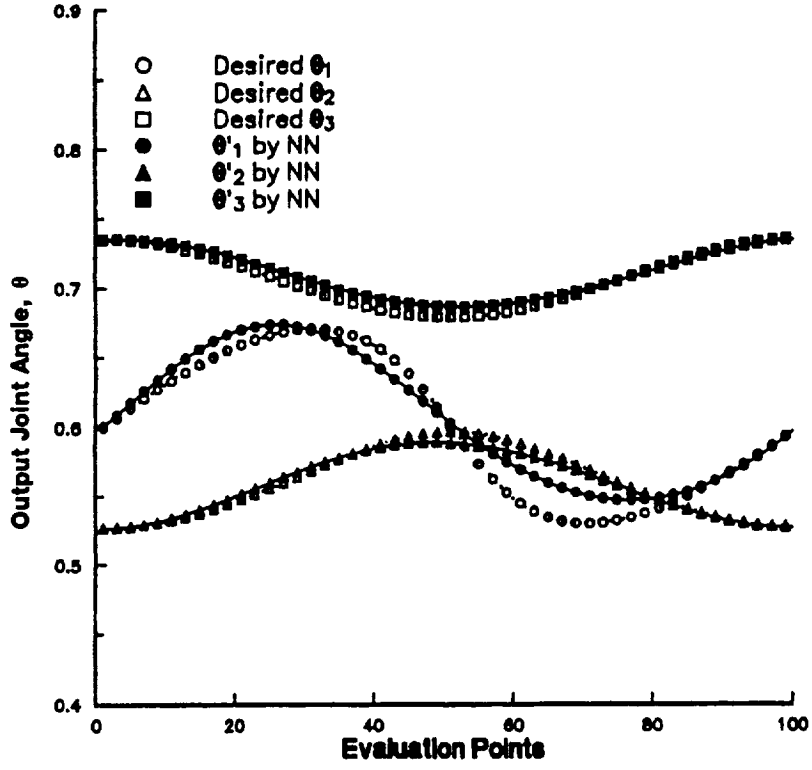


(a) Joint angles.

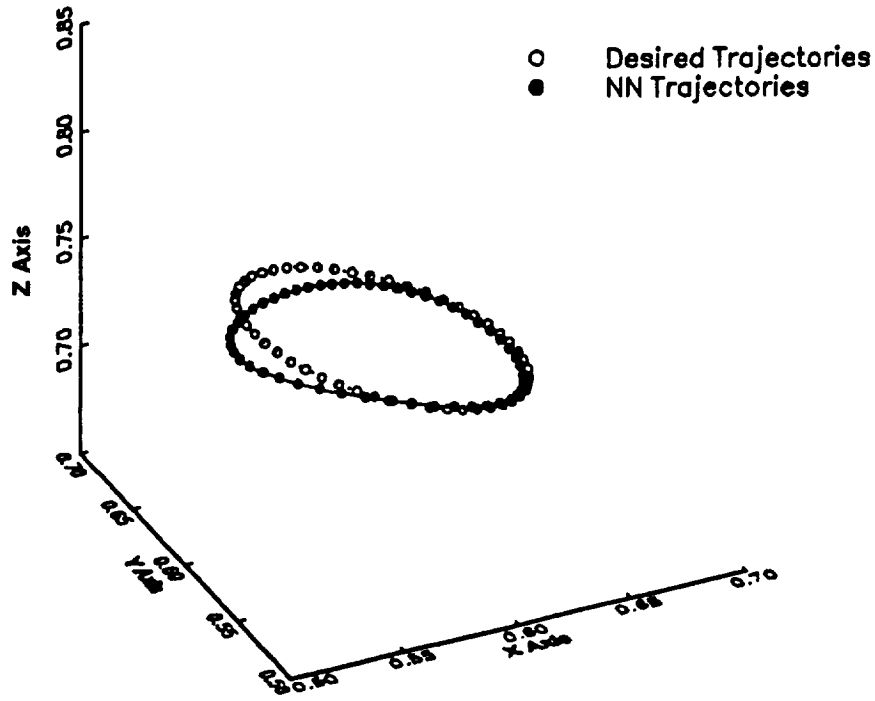


(b) Endeffector trajectory.

Figure 14.—Results from a 3-10-10-3 network using four training points.



(a) Joint angles.



(b) Endeffector trajectories.

Figure 15.—Results from a 3-10-10-3 network using five training points.

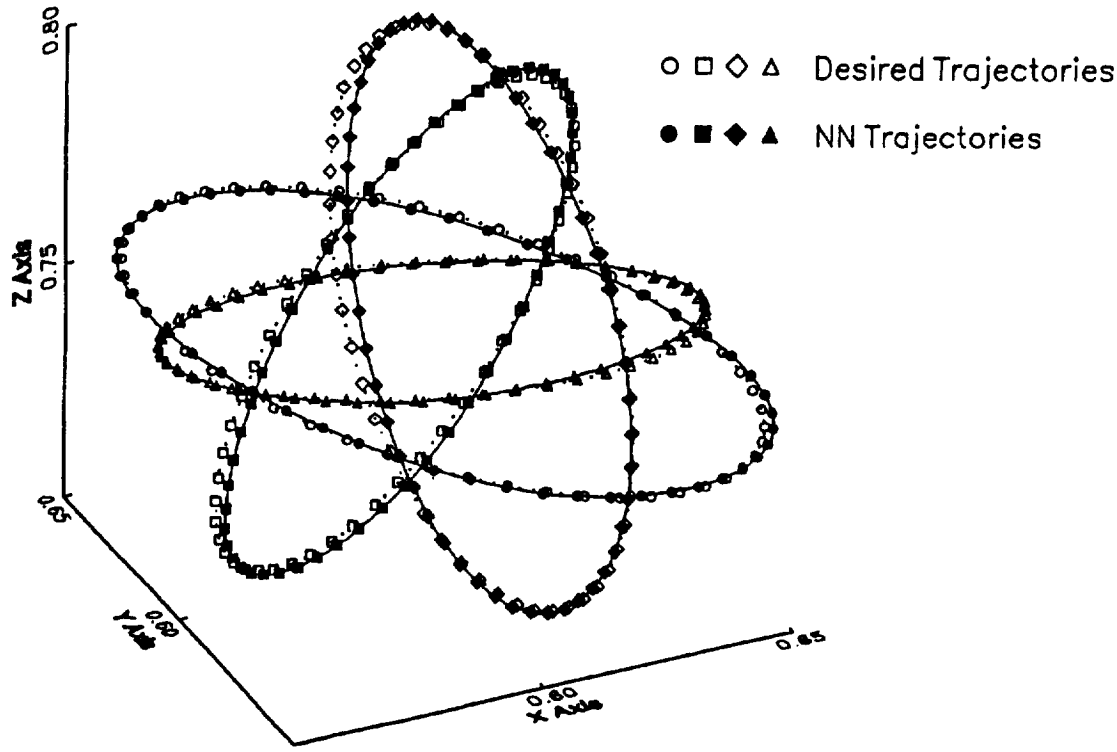


Figure 16.—A 3-10-10-3 network using 27 training points.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1992	Technical Memorandum

4. TITLE AND SUBTITLE

Inverse Kinematics Problem in Robotics Using Neural Networks

5. FUNDING NUMBERS

WU-506-43-41

6. AUTHOR(S)

Benjamin B. Choi and Charles Lawrence

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E-7333

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-105869

11. SUPPLEMENTARY NOTES

Benjamin B. Choi and Charles Lawrence, NASA Lewis Research Center. Responsible person, Benjamin B. Choi, (216) 433-6040.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 39

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

In this paper, Multilayer Feedforward Networks are applied to the robot inverse kinematic problem. The networks are trained with endeffector position and joint angles. After training, performance is measured by having the network generate joint angles for arbitrary endeffector trajectories. A 3-degree-of- freedom (DOF) spatial manipulator is used for the study. It is found that neural networks provide a simple and effective way to both model the manipulator inverse kinematics and circumvent the problems associated with algorithmic solution methods.

14. SUBJECT TERMS

Neural networks; Robotics; Kinematics; Backpropagation; Motion control

15. NUMBER OF PAGES

25

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	